NYT Prediction Report

Exploring Data

There are 8402 observations in total, where 6532 training observations and 1870 observations.

```
popularDensity <- table(newsData$Popular)
posPopular <- round(popularDensity[2]/(popularDensity[1] + popularDensity[2]) * 100, 2)
```

Only 16.73% of all New York Times blog articles have more than 25 comments. That means, a baseline model for predicting unpopular would be around 83.27%.

The independent variables consist of 8 pieces of article data available at the time of publication, and a unique identifier:

- **NewsDesk**, the New York Times desk that produced the story (Business, Culture, Foreign, etc.)
- **SectionName**, the section the article appeared in (Opinion, Arts, Technology, etc.)
- **SubsectionName**, the subsection the article appeared in (Education, Small Business, Room for Debate, etc.)
- **Headline**, the title of the article
- **Snippet**, a small portion of the article text
- **Abstract**, a summary of the blog article, written by the New York Times
- **WordCount**, the number of words in the article
- **PubDate**, the publication date, in the format "Year-Month-Day Hour:Minute:Second"
- **UniqueID**, a unique identifier for each article

Cleaning Data

Text of Articles

Let's take a look at **Headline**, we can observe many very common combination of words like `new york times`, `pictures of the day` etc. If we google `pictures of the day new york times`, it is easy to know `pictures of the day` is a daily article from `Lens` category.

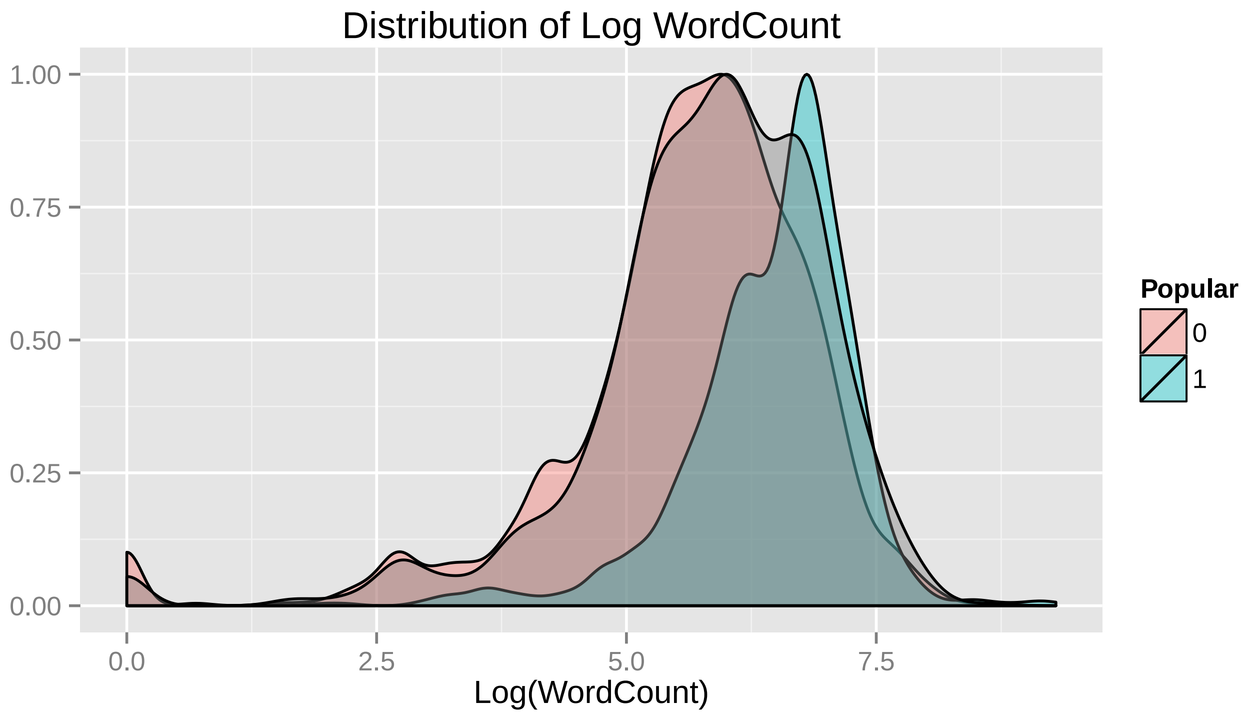
```
newsData$Headline
```

To avoid overcounting of words like `day`, a replacement of some proper nouns to single word is necessary.

```
originalText    = c("new york times", "new york city", "new york", "silicon valley",
                    "times insider", "fashion week", "white house",
                    "international herald tribune archive",
                    "president obama", "hong kong", "big data", "golden globe",
                    "word of the day", "time square", "pictures of the day",
                    "photos of the day", "daily clip report")

replacementText = c("NYT", "NYC", "NewYork", "SiliconValley", "TimesInsider",
                    "FashionWeek", "WhiteHouse", "IHT", "Obama", "HongKong",
                    "BigData", "GoldenGlobe", "WordofDay", "TimeSquare", "PicOfDay",
                    "PicOfDay", "DailyClipReport")
```

Word Count of Articles

The following plot shows article's popularity distribution based on logarithmic word count. Beside training data, testing data's distribution is also plotted by gray color which is bimodal distribution.



If we conduct a two-sided t-test on the mean and a two-sided F-test on the variance:

```
PopularNewsTrain   = subset(newsTrain, newsTrain$Popular==1)
UnpopularNewsTrain = subset(newsTrain, newsTrain$Popular==0)

t.test(PopularNewsTrain$LogWordCount, UnpopularNewsTrain$LogWordCount)
```

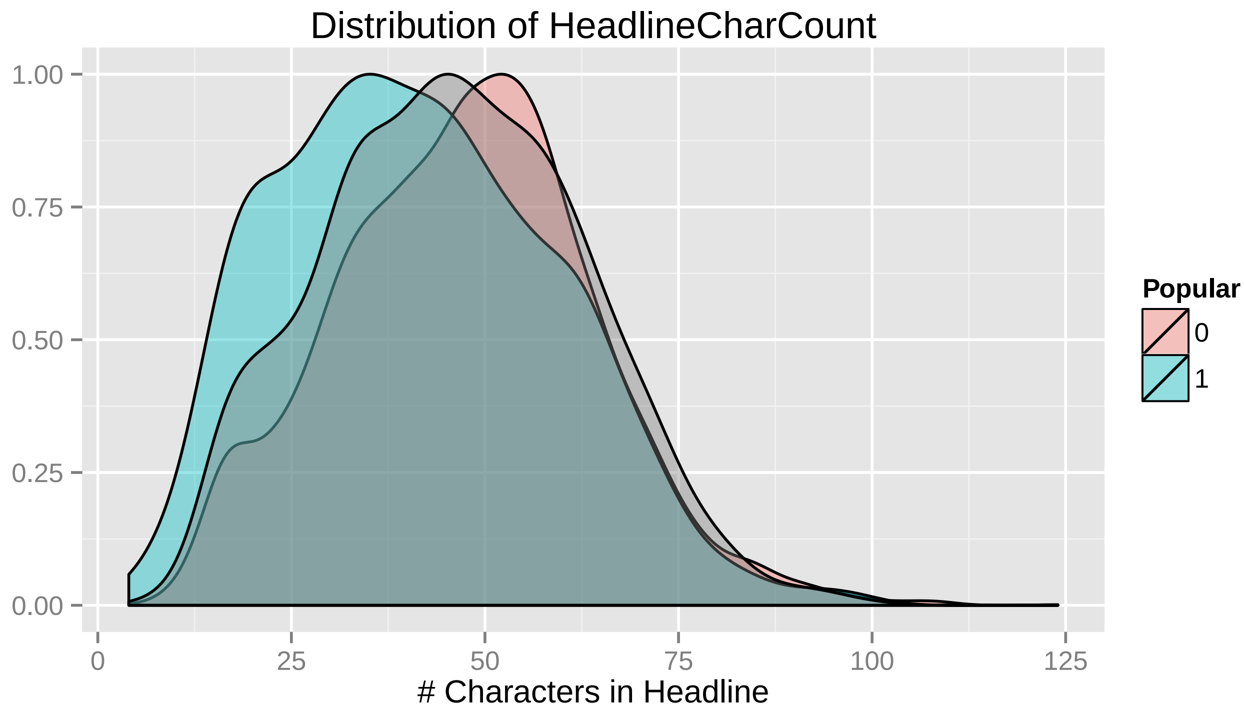
```
##
##  Welch Two Sample t-test
##
## data:  PopularNewsTrain$LogWordCount and UnpopularNewsTrain$LogWordCount
## t = 28.2691, df = 2310.719, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.8018917 0.9214367
## sample estimates:
## mean of x mean of y
##  6.455548  5.593884
```

```
var.test(PopularNewsTrain$LogWordCount, UnpopularNewsTrain$LogWordCount)
```

```
##
##  F test to compare two variances
##
```

```
## data:  PopularNewsTrain$LogWordCount and UnpopularNewsTrain$LogWordCount
## F = 0.4108, num df = 1092, denom df = 5438, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.3752262 0.4509859
## sample estimates:
## ratio of variances
##          0.4107796
```

This shows us there is a statistically significant difference between popular and unpopular articles based on the word counts. At the same time, popular article seems having shorter `Headline`.




```
t.test(PopularNewsTrain$HeadlineCharCount, UnpopularNewsTrain$HeadlineCharCount)
```

```
##
##  Welch Two Sample t-test
##
## data:  PopularNewsTrain$HeadlineCharCount and UnpopularNewsTrain$HeadlineCharCount
## t = -10.8261, df = 1494.655, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -7.417725 -5.142058
## sample estimates:
## mean of x mean of y
##  41.16651  47.44641
```

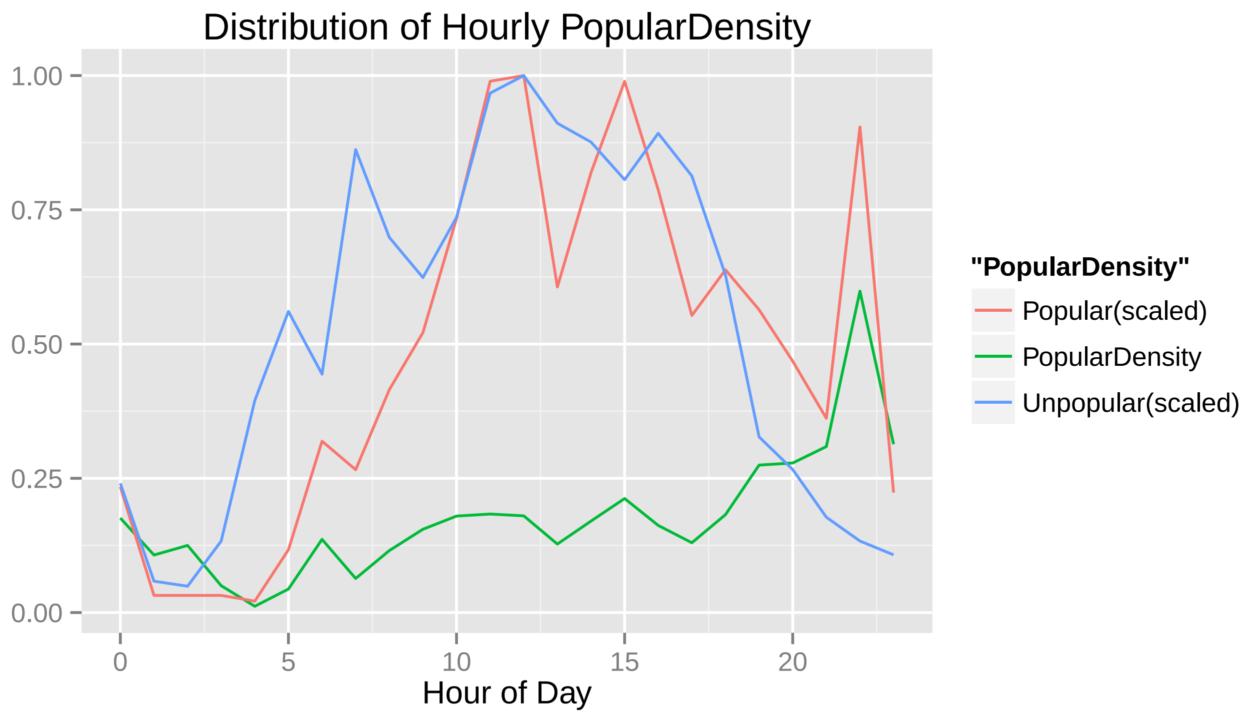
Publishing Hour and Day

It is unlikely that many article receiving 25 more comments in the middle of night. Hence, at certain times during the day, we expect the probability that a random article becomes popular to be larger. Similarly, the day of the week may have an impact, people may have much more time to read articles than a working day.

```r
## date feature
newsData$PubDate = strptime(newsData$PubDate, "%Y-%m-%d %H:%M:%S")
newsData$PubDay  = as.Date(newsData$PubDate)
## it is expected that different behaviours at different times of the day.publication
newsData$DayofWeek = newsData$PubDate$wday
newsData$Hour    = newsData$PubDate$hour
newsData$DayOfWeek = as.factor(weekdays(newsData$PubDate))
newsData$DayOfWeek = factor(newsData$DayOfWeek, levels=c("Monday", "Tuesday", "Wednesday", "Thursday",

## especially on holidays, people may have much more time to read and comment on blog articles
Holidays = c(as.POSIXlt("2014-09-01 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-10-13 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-10-31 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-11-11 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-11-27 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-12-24 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-12-25 00:00", format="%Y-%m-%d %H:%M"),
             as.POSIXlt("2014-12-31 00:00", format="%Y-%m-%d %H:%M"))

newsData$Holiday = as.factor(ifelse(newsData$PubDate$yday %in% Holidays$yday, 1, 0))
```

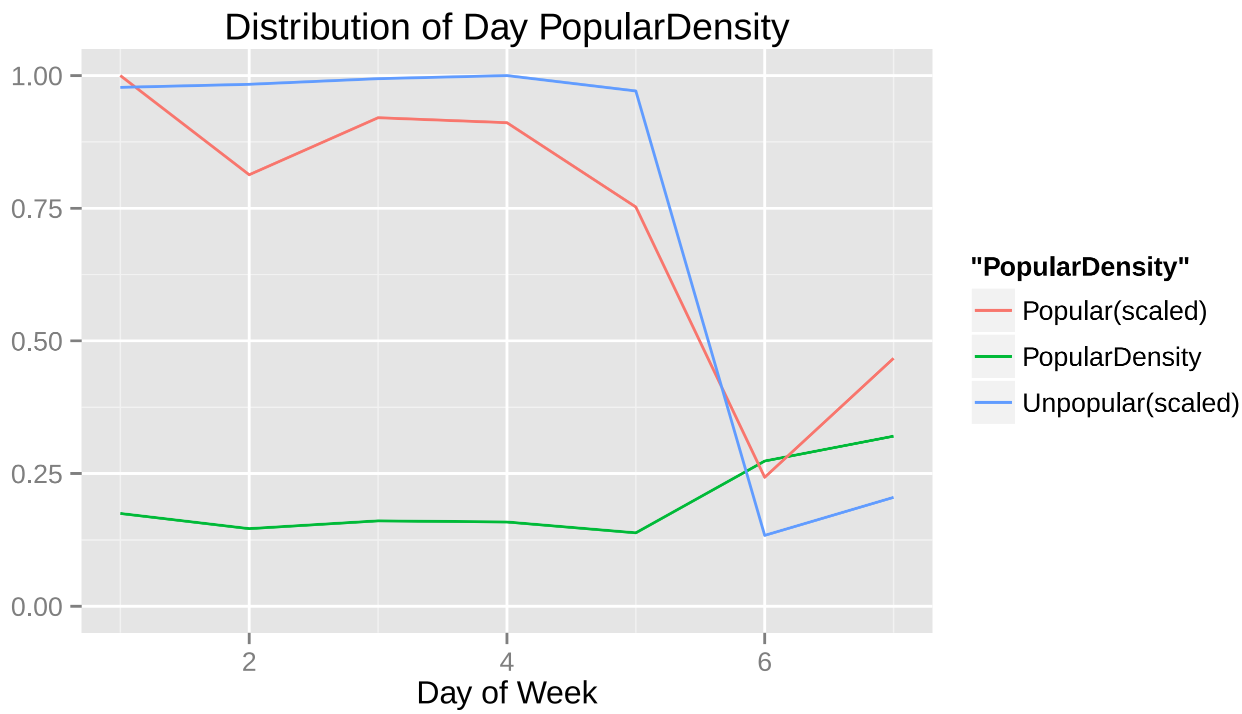
```
##    Unpopular Popular PopularDensity Hour
## 4        169       2     0.01169591    4
## 5        240      11     0.04382470    5
## 3         57       3     0.05000000    3
## 7        369      25     0.06345178    7
## 1         25       3     0.10714286    1
## 8        299      39     0.11538462    8
## 2         21       3     0.12500000    2
## 13       390      57     0.12751678   13
## 17       348      52     0.13000000   17
## 6        190      30     0.13636364    6
## 9        267      49     0.15506329    9
## 16       382      74     0.16228070   16
## 14       375      77     0.17035398   14
## 0        103      22     0.17600000    0
## 10       315      69     0.17968750   10
## 12       428      94     0.18007663   12
## 18       269      60     0.18237082   18
## 11       414      93     0.18343195   11
## 15       345      93     0.21232877   15
## 19       140      53     0.27461140   19
## 20       114      44     0.27848101   20
## 21        76      34     0.30909091   21
## 23        46      21     0.31343284   23
## 22        57      85     0.59859155   22
```

It seems that publishing blog posts around `10 pm` are more easier getting popular according to `PopularDensity`. But, if we compare number of popular articles of 24 hours, It is clear to see that around `12 pm` and `3 pm`, even more blog posts receiving 25 more comments than `10 pm`.




```
##           Unpopular Popular PopularDensity Day
## Friday         1003     161      0.1383162   5
## Tuesday        1016     174      0.1462185   2
## Thursday       1033     195      0.1587948   4
## Wednesday      1027     197      0.1609477   3
## Monday         1010     214      0.1748366   1
## Saturday        138      52      0.2736842   6
## Sunday          212     100      0.3205128   7
```

Similarly, day of week shows same trends as hourly results. Also, much more articles are published on weekday than weekends.



Category of Articles

There are three variables categorizes blog posts, `NewsDesk` `SectionName` and `SubsectionName`.

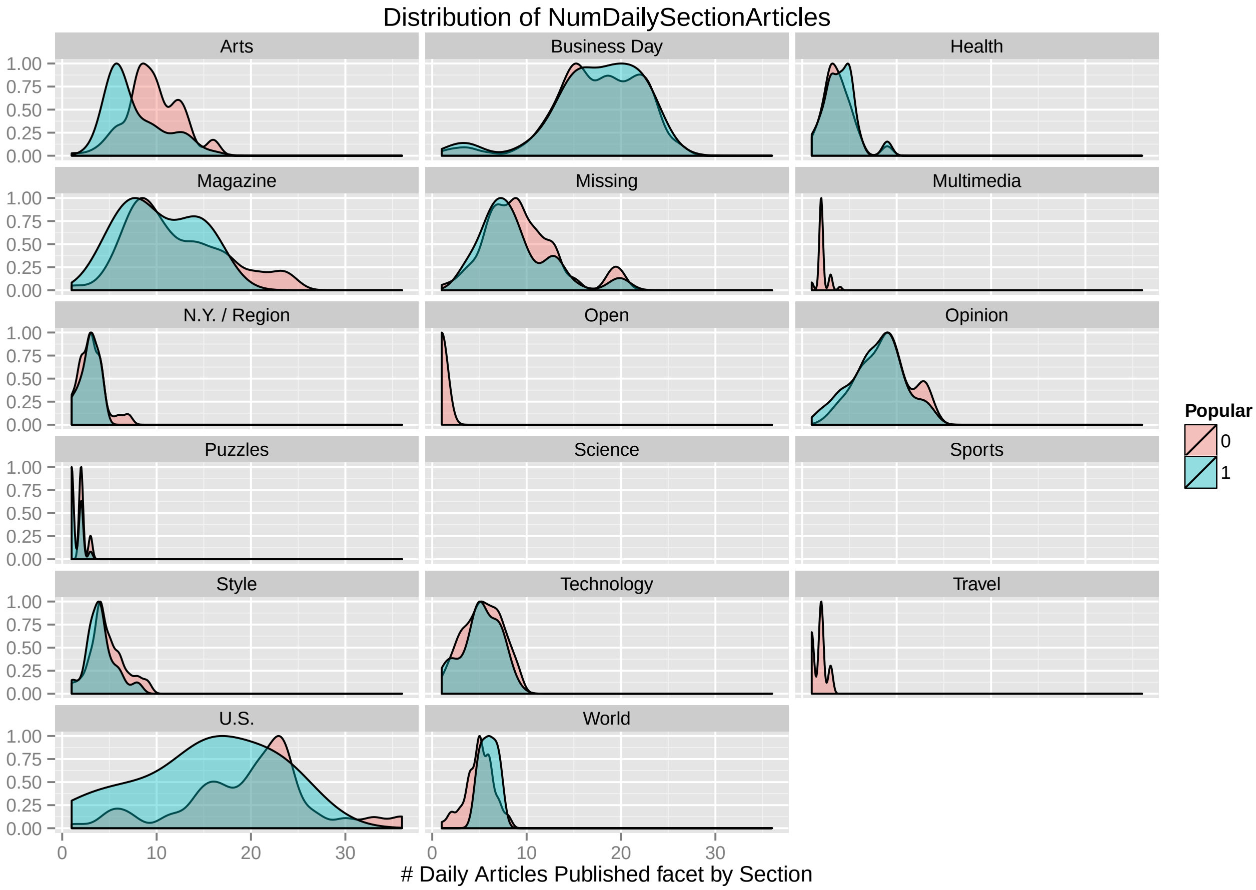
```
## missing categories
misCategory = subset(newsData, newsData$NewsDesk=="" | newsData$SectionName=="" | newsData$SubsectionNa
dim(misCategory)[1]
```

```
## [1] 6721
```

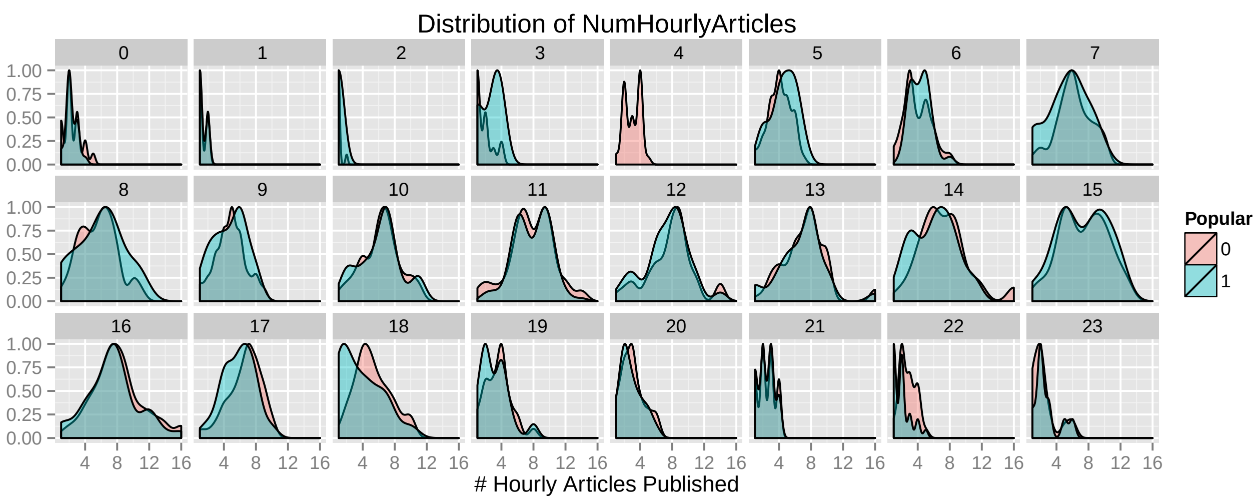
```
misCategory = subset(newsData, newsData$NewsDesk=="" & newsData$SectionName=="" & newsData$SubsectionNa
dim(misCategory)[1]
```

```
## [1] 1626
```

6721 articles have at one category variable missing and **1626** articles have no categories at all. After filling blank categories based on existing category variables, let's try to see the facet distribution of blog posts.



Although it is hard to see what's going on, a clear difference between popular and unpopular articles is in the section Magazine, where around 15 articles posted per day is more indicative of popular articles than unpopular ones. Beyond 20 posts per day the roles are clearly reversed. Hourly distribution of the following plot also shows no clear indication of popularity.

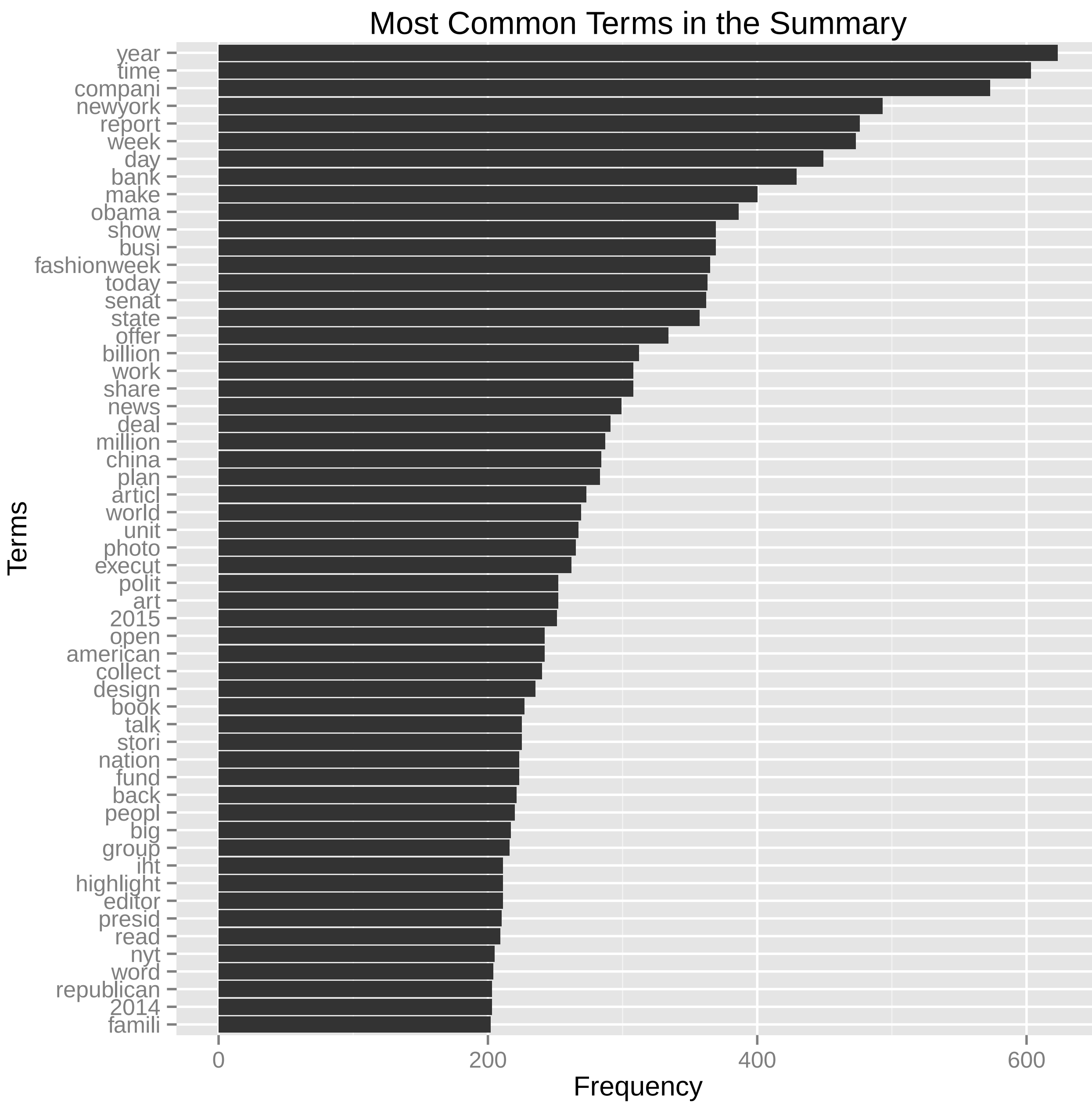


Contents Features of Articles

Until now, we preprocessed date features, word counts and categories of article. Almost all features from original data frame but the contents of blog post.

```r
stopWords = c(stopwords("SMART"))
CorpusText = Corpus(VectorSource(newsData$Text))
CorpusText = tm_map(CorpusText, tolower)
CorpusText = tm_map(CorpusText, PlainTextDocument)
CorpusText = tm_map(CorpusText, removePunctuation)
CorpusText = tm_map(CorpusText, removeWords, stopWords)
CorpusText = tm_map(CorpusText, stemDocument, language="english")

tdmText = TermDocumentMatrix(CorpusText)
sparseText = removeSparseTerms(tdmText, 0.98)
sparseText = as.data.frame(as.matrix(sparseText))
colnames(sparseText) = make.names(colnames(sparseText))

dtmText = DocumentTermMatrix(CorpusText)
freqTerms = findFreqTerms(dtmText, lowfreq=10)
termFreq  = colSums(as.matrix(dtmText))
termFreq  = subset(termFreq, termFreq>=200)
df        = data.frame(term=names(termFreq), freq=termFreq)

newsDataNoBagWords = newsData

tSparseText = t(sparseText)
colnames(tSparseText) = make.names(paste('c',colnames(tSparseText)))
newsData[, colnames(tSparseText)] = tSparseText
```

```r
ggplot(df, aes(x=reorder(term, freq, max), y=freq)) +
  geom_bar(stat="identity") +
  ggtitle("Most Common Terms in the Summary") +
  xlab("Terms") +
  ylab("Frequency") +
  coord_flip()
```



Modeling Data

Logistic Regression without contents feature

```
lrModel = glm(Popular ~ PubDay + Hour +
                        WordCount + DayofWeek + HeadlineCharCount + SummaryCharCount +
                        HeadlineWordCount + SummaryWordCount + LogWordCount +
                        NumDailyArticles + NumDailySectionArticles + NumHourlyArticles +
                        ShortHeadline + Holiday,
              data=newsTrain, family=binomial)
calcAUClr(lrModel, newsTrain$Popular)
```

```
## [1] 0.8553276 0.8281058
```

```
lrModelPred = predict(lrModel, newdata=newsTest, type="response")
generateSubmission(lrModelPred)
```

Random Forest with contents feature

```
# modeling
## random forest
rfModel = randomForest(Popular ~ PubDay + Hour +
                                 WordCount + DayofWeek + HeadlineCharCount + SummaryCharCount +
                                 HeadlineWordCount + SummaryWordCount + LogWordCount +
                                 NumDailyArticles + NumDailySectionArticles + NumHourlyArticles +
                                 ShortHeadline + Holiday,
                      data=newsTrain, nodesize=5, ntree=1000, importance=TRUE)

trainPartition = createDataPartition(y=newsTrain$Popular, p=0.5, list=FALSE)
tuneTrain      = newsTrain[trainPartition, ]
rfModel.tuned  = train(Popular ~ PubDay + Hour +
                                 WordCount + DayofWeek + HeadlineCharCount + SummaryCharCount +
                                 HeadlineWordCount + SummaryWordCount + LogWordCount +
                                 NumDailyArticles + NumDailySectionArticles + NumHourlyArticles +
                                 ShortHeadline + Holiday,
                      data=tuneTrain,
                      method="rf",
                      trControl=trainControl(method="cv", number=5))
calcAUC(rfModel, newsTrain$Popular)
```

```
## [1] 0.8727802 0.8752473
```

```
rfModelPred = predict(rfModel, newdata=newsTest, type="prob")[,2]
generateSubmission(rfModelPred)
```

Logistic Regression

```
removedColumns = c("SectionName", "NewsDesk", "SubsectionName", "Headline", "Snippet", "Abstract", "Summ
lrModelText = glm(Popular ~ ., data=newsTrain[,!colnames(newsTrain) %in% removedColumns], family=binomi
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
calcAUClr(lrModelText, newsTrain$Popular)
```

```
## [1] 0.8649724 0.8731482
```

```
lrModelTextPred = predict(lrModelText, newdata=newsTest, type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```r
generateSubmission(lrModelTextPred)
```

Random Forest

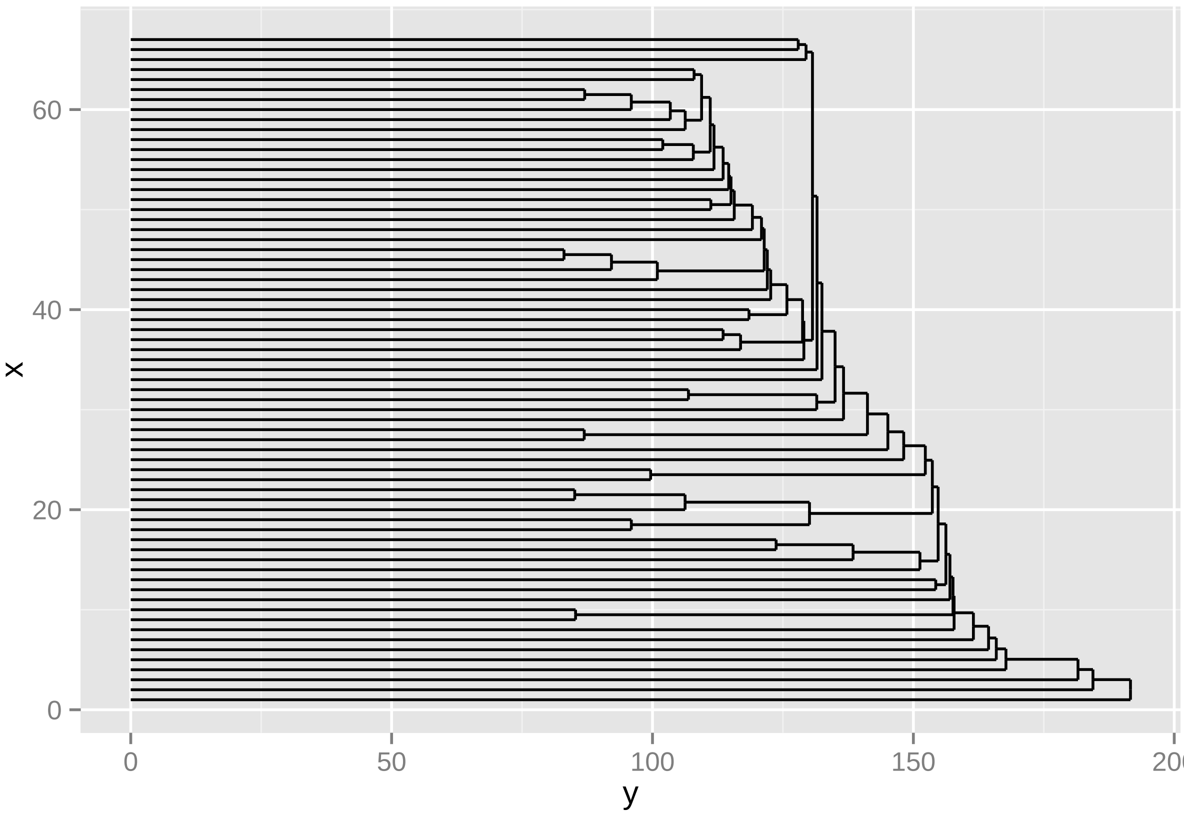
```r
newsTrain = na.omit(newsTrain)
rfModelText = randomForest(Popular ~ . -SectionName
                                       -NewsDesk
                                       -SubsectionName
                                       -Headline
                                       -Snippet
                                       -Abstract
                                       -Summary
                                       -UniqueID
                                       -Text,
                           data=newsTrain, nodesize=5, ntree=1000, importance=TRUE)

trainPartition = createDataPartition(y=newsTrain$Popular, p=0.5, list=FALSE)
tuneTrain      = newsTrain[trainPartition, ]
rfModelText.tuned  = train(Popular ~ . -SectionName
                                       -NewsDesk
                                       -SubsectionName
                                       -Headline
                                       -Snippet
                                       -Abstract
                                       -Summary
                                       -UniqueID
                                       -Text,
                          data=tuneTrain,
                          method="rf",
                          trControl=trainControl(method="cv", number=5))
calcAUC(rfModelText, newsTrain$Popular)
```

```
## [1] 1 1
```

```r
rfModelTextPred = predict(rfModelText, newdata=newsTest, type="prob")[,2]
generateSubmission(rfModelTextPred)
```

Supervised + Unsupervised

```r
matrixSparseText   = as.matrix(sparseText)
matrixSparseText.distMatrix = dist(scale(matrixSparseText))
matrixSparseText.clusters   = hclust(matrixSparseText.distMatrix, method="ward.D2")

dText = as.dendrogram(matrixSparseText.clusters)
dTextData <- dendro_data(dText, type = "rectangle")

ggplot(segment(dTextData)) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend)) +
  coord_flip()
```




```
kText   = 25
mText   = t(sparseText)
KMCText = kmeans(mText, kText)

for (i in 1:kText) {
  cat(paste("cluster ", i, ": ", sep=","))
  s = sort(KMCText$centers[i, ], decreasing=TRUE)
  cat(names(s)[1:15], sep=", ", "\n")
}
```

```
## cluster ,1,: obama, presid, polit, make, today, nation, call, plan, report, state, show, execut, time
## cluster ,2,: newyork, today, day, citi, show, fashionweek, open, art, week, world, year, back, diari
## cluster ,3,: day, iht, make, show, work, china, peopl, share, polit, art, world, plan, highlight, pr
## cluster ,4,: time, articl, collect, execut, media, day, manag, morn, editor, good, 2014, 2015, ameri
## cluster ,5,: bank, big, billion, morn, million, newyork, plan, back, compani, year, execut, report,
## cluster ,6,: republican, senat, polit, today, obama, nation, state, group, big, day, back, deal, tal
## cluster ,7,: state, unit, nation, china, 2014, photo, presid, report, world, obama, american, make,
## cluster ,8,: time, report, 2014, offer, discuss, share, photo, week, articl, famili, stori, talk, sh
## cluster ,9,: fashionweek, 2015, diari, newyork, photo, collect, day, show, morn, report, editor, cit
## cluster ,10,: year, back, day, make, report, time, art, 2014, plan, work, busi, china, million, offe
## cluster ,11,: week, art, open, show, world, famili, share, take, news, includ, day, highlight, stori
## cluster ,12,: compani, market, execut, make, million, plan, share, report, year, china, big, billion
## cluster ,13,: book, talk, discuss, nation, editor, time, 2014, world, make, nyt, stori, collect, day
## cluster ,14,: design, collect, 2015, art, fashionweek, open, show, newyork, work, year, world, big,
## cluster ,15,: billion, offer, compani, share, rais, million, group, famili, make, plan, fund, manag,
## cluster ,16,: fund, billion, rais, manag, newyork, bank, million, state, compani, offer, plan, world
## cluster ,17,: fund, manag, million, rais, market, morn, offer, billion, deal, group, plan, work, com
```

```
## cluster ,18,: week, news, 2014, morn, year, includ, stori, 2015, american, art, articl, back, bank, b
## cluster ,19,: report, stori, nyt, editor, highlight, today, time, china, morn, day, news, week, make
## cluster ,20,: american, million, day, newyork, china, open, show, year, make, art, includ, state, wee
## cluster ,21,: deal, billion, compani, group, year, big, busi, make, unit, million, offer, talk, inclu
## cluster ,22,: news, editor, week, time, media, stori, report, famili, nation, peopl, china, good, wor
## cluster ,23,: word, articl, year, nyt, make, back, call, nation, 2014, american, art, big, book, deal
## cluster ,24,: senat, polit, republican, day, obama, show, state, report, today, take, make, presid, g
## cluster ,25,: busi, today, compani, market, plan, offer, big, execut, million, group, make, year, ra
```

```r
newsData$TextCluster     = as.factor(KMCText$cluster)
newsDataNoBagWords$PubDate = NULL
newsDataNoBagWords$TextCluster = newsData$TextCluster
newsTrain = head(newsDataNoBagWords, nrow(trainData))
newsTest  = tail(newsDataNoBagWords, nrow(testData))
```

```r
rfModelMix = randomForest(Popular ~ PubDay + Hour + TextCluster +
                                 WordCount + DayofWeek + HeadlineCharCount + SummaryCharCount +
                                 HeadlineWordCount + SummaryWordCount + LogWordCount +
                                 NumDailyArticles + NumDailySectionArticles + NumHourlyArticles +
                                 ShortHeadline + Holiday,
                           data=newsTrain, nodesize=5, ntree=1000, importance=TRUE)

trainPartition = createDataPartition(y=newsTrain$Popular, p=0.5, list=FALSE)
tuneTrain      = newsTrain[trainPartition, ]
rfModelMix.tuned  = train(Popular ~ PubDay + Hour + TextCluster +
                                 WordCount + DayofWeek + HeadlineCharCount + SummaryCharCount +
                                 HeadlineWordCount + SummaryWordCount + LogWordCount +
                                 NumDailyArticles + NumDailySectionArticles + NumHourlyArticles +
                                 ShortHeadline + Holiday,
                          data=tuneTrain,
                          method="rf",
                          trControl=trainControl(method="cv", number=5))
calcAUC(rfModelMix, newsTrain$Popular)
```

```
## [1] 0.8724740 0.8852602
```

```r
rfModelMixPred = predict(rfModelMix, newdata=newsTest, type="prob")[,2]
generateSubmission(rfModelMixPred)
```